

(19) 日本国特許庁(JP)

(12) 特 許 公 報(B2)

(11) 特許番号

特許第4688111号

(P4688111)

(45) 発行日 平成23年5月25日(2011.5.25)

(24) 登録日 平成23年2月25日(2011.2.25)

(51) Int.Cl.

F I

G O 6 F 17/30 (2006.01)

G O 6 F 17/30 4 1 4 A

G O 6 F 12/00 (2006.01)

G O 6 F 12/00 5 4 7 Z

G O 6 F 17/30 1 4 O

G O 6 F 17/30 2 3 O Z

請求項の数 20 (全 27 頁)

(21) 出願番号 特願2008-305698 (P2008-305698)
 (22) 出願日 平成20年11月28日(2008.11.28)
 (65) 公開番号 特開2010-129001 (P2010-129001A)
 (43) 公開日 平成22年6月10日(2010.6.10)
 審査請求日 平成22年10月28日(2010.10.28)

(73) 特許権者 390009531
 インターナショナル・ビジネス・マシーンズ・コーポレーション
 INTERNATIONAL BUSINESS MACHINES CORPORATION
 アメリカ合衆国10504 ニューヨーク州 アーモンク ニュー オーチャードロード
 (74) 代理人 100108501
 弁理士 上野 剛史
 (74) 代理人 100112690
 弁理士 太佐 種一
 (74) 代理人 100091568
 弁理士 市位 嘉宏

最終頁に続く

(54) 【発明の名称】 情報処理装置、データベース・システム、情報処理方法、およびプログラム

(57) 【特許請求の範囲】

【請求項 1】

階層構造を有する構造化文書を管理するデータベースを含む情報処理装置であって、前記情報処理装置は、

前記構造化文書のパーティション化する対象データを含む単位部分木を、前記対象データの値に応答して分離して登録する値パーティションと、前記単位部分木以外の部分木および前記単位部分木を固有に識別するためのインデックスを含むベース・パーティションとを生成するパーティション生成部と、

前記構造化文書の属性を定義する開始タグの位置および終了タグの位置を識別して、属性値の階層構造に対応する分割位置リストを作成し記憶領域に登録するパーザ部と、

前記分割位置リストを参照し、前記対象データを含む分割するべき前記単位部分木の前記開始タグ位置および前記終了タグ位置を識別して前記単位部分木および前記単位部分木を参照するためのインデックスを、前記構造化文書から分離する分割部と、

前記分割部により分離した前記単位部分木と前記単位部分木を識別するインデックスとを異なるパーティションに登録する登録部と

を含む情報処理装置。

【請求項 2】

前記単位部分木は、複数の前記開始タグを含み、前記インデックスは、前記単位部分木の最初の前記開始タグから生成される、請求項 1 に記載の情報処理装置。

【請求項 3】

10

20

前記登録部は、前記単位部分木と、前記単位部分木を固有に識別するための前記インデックスとに対し、相互参照を可能とするポインタをそれぞれ追加して前記異なるパーティションに登録する、請求項2に記載の情報処理装置。

【請求項4】

前記分割部は、前記分割位置リストを参照し、前記異なるパーティションに登録するための前記単位部分木を分離するため、前記構造化文書が含む前記単位部分木の開始位置および終了位置とを対応付ける、領域ペアおよび当該単位部分木の前記構造化文書内での記述順に登録する出現カウンタを含む部分木データ・リストを作成し、記憶領域に登録する、請求項3に記載の情報処理装置。

【請求項5】

前記情報処理装置は、前記値パーティションに登録された前記単位部分木を前記対象データを使用して検索するためのSQL処理部と、前記SQL処理部による検索結果を生成する検索結果生成部を含む、前記請求項4に記載の情報処理装置。

【請求項6】

前記構造化文書は、XMLで記述され、前記データベースは、XMLを格納・検索することを可能にするXMLDBである、請求項4に記載の情報処理装置。

【請求項7】

階層構造を有する構造化文書に登録するデータベースを含むデータベース・システムであって、前記データベース・システムは、

前記構造化文書のパーティション化する対象データを含む単位部分木を、前記対象データの値に応答して分離して登録する値パーティションと、前記単位部分木以外の部分木および前記単位部分木を固有に識別するためのインデックスを含むベース・パーティションとを生成して管理するデータ管理部と、

前記データベースに登録された前記構造化文書を、当該構造化文書の前記階層構造を指定した検索要求を処理するSQL処理部と、

前記SQL処理部による前記値パーティションについての個別検索結果から前記検索要求に対する検索結果を作成する検索結果作成部と

を含み、前記データ管理部は、

前記パーティションを生成するパーティション生成部と、

前記構造化文書の属性を定義する開始タグの位置および終了タグの位置を識別して、属性値の階層構造に対応する分割位置リストを作成し記憶領域に登録するパーザ部と、

前記分割位置リストを参照し、前記対象データを含む分割すべき前記単位部分木の前記開始タグ位置および前記終了タグ位置を識別して前記単位部分木および前記単位部分木を参照するためのインデックスを、前記構造化文書から分離する分割部と、

前記分割部により分離した前記単位部分木と前記単位部分木を識別するインデックスとを異なるパーティションに登録する登録部と

を含む、データベース・システム。

【請求項8】

前記単位部分木は、複数の前記開始タグを含み、前記インデックスは、前記単位部分木の最初の前記開始タグから生成される、請求項7に記載のデータベース・システム。

【請求項9】

前記登録部は、前記単位部分木と、前記単位部分木を固有に識別するための前記インデックスとに対し、相互参照を可能とするポインタをそれぞれ追加して前記異なるパーティションに登録する、請求項8に記載のデータベース・システム。

【請求項10】

前記分割部は、前記分割位置リストを参照し、前記異なるパーティションに登録するための前記単位部分木を分離するため、前記構造化文書が含む前記単位部分木の開始位置および終了位置とを対応付ける、領域ペアおよび当該単位部分木の前記構造化文書内での記述順に登録する出現カウンタを含む部分木データ・リストを作成し、記憶領域に登録する、請求項9に記載のデータベース・システム。

10

20

30

40

50

【請求項 1 1】

前記構造化文書は、XMLで記述され、前記データベースは、XMLを格納・検索することを可能にするXMLDBシステムである、請求項 1 0 に記載のデータベース・システム。

【請求項 1 2】

階層構造を有する構造化文書を登録するデータベースに対する情報処理方法であって、前記情報処理方法は、情報処理装置が、

前記構造化文書のパーティション化する対象データを含む単位部分木を、前記対象データの値に応答して分離して登録する値パーティションと、前記単位部分木以外の部分木および前記単位部分木を固有に識別するためのインデックスを含むベース・パーティションとを生成するステップと、

10

前記構造化文書の属性を定義する開始タグの位置および終了タグの位置を識別して、属性値の階層構造に対応する分割位置リストを作成し記憶領域に登録するステップと、

前記分割位置リストを参照し、前記対象データを含む分割すべき前記単位部分木の前記開始タグ位置および前記終了タグ位置を識別して前記単位部分木および前記単位部分木を参照するためのインデックスを、前記構造化文書から分離するステップと

前記分離するステップで分離された前記単位部分木と前記単位部分木を識別するインデックスとを異なるパーティションに登録するステップと

を実行する、情報処理方法。

【請求項 1 3】

前記単位部分木は、複数の前記開始タグを含み、前記生成するステップは、前記インデックスを、前記単位部分木の最初の前記開始タグを識別して生成するステップを含む、請求項 1 2 に記載の情報処理方法。

20

【請求項 1 4】

前記分離するステップは、前記分割位置リストを参照し、前記異なるパーティションに登録するための前記単位部分木を分離するため、前記構造化文書が含む前記単位部分木の開始位置および終了位置とを対応付ける、領域ペアおよび当該単位部分木の前記構造化文書内での記述順を登録する出現カウンタを含む部分木データ・リストを作成し、記憶領域に登録するステップを含む、請求項 1 3 に記載の情報処理方法。

【請求項 1 5】

前記登録するステップは、前記単位部分木と、前記単位部分木を固有に識別するための前記インデックスとに対し、相互参照を可能とするポインタをそれぞれ追加して前記異なるパーティションに登録するステップを含む、請求項 1 4 に記載の情報処理方法。

30

【請求項 1 6】

さらに、

前記値パーティションに登録された前記単位部分木を前記対象データを使用して検索するステップと、

前記検索するステップによる前記値パーティションについての個別検索結果を統合して検索結果を生成するステップと

を含む、請求項 1 5 に記載の情報処理方法。

【請求項 1 7】

40

階層構造を有する構造化文書を登録するデータベースに対する情報処理方法を情報処理装置が実行するためのコンピュータ実行可能なプログラムであって、前記プログラムは、情報処理装置が、

前記構造化文書のパーティション化する対象データを含む単位部分木を、前記対象データの値に応答して分離して登録する値パーティションと、前記単位部分木以外の部分木および前記単位部分木を固有に識別するためのインデックスを含むベース・パーティションとを生成するステップと、

前記構造化文書の属性を定義する開始タグの位置および終了タグの位置を識別して、属性値の階層構造に対応する分割位置リストを作成し記憶領域に登録するステップと、

前記分割位置リストを参照し、前記対象データを含む分割すべき前記単位部分木の前

50

記開始タグ位置および前記終了タグ位置を識別して前記単位部分木および前記単位部分木を参照するためのインデックスを、前記構造化文書から分離するステップと

前記分離するステップで分離された前記単位部分木と前記単位部分木を識別するインデックスとを異なるパーティションに登録するステップと、
を実行するための、コンピュータ実行可能なプログラム。

【請求項 18】

前記単位部分木は、複数の前記開始タグを含み、前記生成するステップは、前記インデックスを、前記単位部分木の最初の前記開始タグを識別して生成するステップを含む、請求項 17 に記載のプログラム。

【請求項 19】

前記分離するステップは、前記分割位置リストを参照し、前記異なるパーティションに登録するための前記単位部分木を分離するため、前記構造化文書が含む前記単位部分木の開始位置および終了位置とを対応付ける、領域ペアおよび当該単位部分木の前記構造化文書内での記述順に登録する出現カウンタを含む部分木データ・リストを作成し、記憶領域に登録するステップを含む、請求項 18 に記載のプログラム。

【請求項 20】

階層構造を有する構造化文書を管理するデータベースを含む情報処理装置であって、前記情報処理装置は、

前記構造化文書のパーティション化する対象データを含む単位部分木を、前記対象データの値に応答して分離して登録する値パーティションと、前記単位部分木以外の部分木および前記単位部分木を固有に識別するためのインデックスを含むベース・パーティションとを生成するパーティション生成部と、

前記構造化文書の属性を定義する開始タグの位置および終了タグの位置を識別して、属性値の階層構造に対応する分割位置リストを作成し記憶領域に登録するパーザ部と、

前記分割位置リストを参照し、前記対象データを含む分割すべき前記単位部分木の前記開始タグ位置および前記終了タグ位置を識別して前記単位部分木および前記単位部分木を参照するためのインデックスを、前記構造化文書から分離する分割部と、

前記分割部により分離した前記単位部分木と前記単位部分木を識別するインデックスとを異なるパーティションに登録する登録部と

を含み、

前記単位部分木は、複数の前記開始タグを含み、前記インデックスは、前記単位部分木の最初の前記開始タグから生成され、

前記登録部は、前記単位部分木と、前記単位部分木を固有に識別するための前記インデックスとに対し、相互参照を可能とするポインタをそれぞれ追加して前記異なるパーティションに登録し、

前記分割部は、さらに前記分割位置リストを参照し、前記異なるパーティションに登録するための前記単位部分木を分離するため、前記構造化文書が含む前記単位部分木の開始位置および終了位置とを対応付ける、領域ペアおよび当該単位部分木の前記構造化文書内での記述順に登録する出現カウンタを含む部分木データ・リストを作成し、記憶領域に登録しており、前記構造化文書は、XMLで記述され、前記データベースは、XMLを格納・検索することを可能にするXMLDBである、情報処理装置。

【発明の詳細な説明】

【技術分野】

【0001】

本発明は、データベースを対象とする情報処理技術に関し、より詳細には、データベースのデータ領域をパーティションすることによりデータベース・アクセスを効率化する、情報処理装置、データベース・システム、情報処理方法およびプログラムに関する。

【背景技術】

【0002】

インターネットといったネットワーク基盤の普及および情報処理装置やアプリケーション

10

20

30

40

50

ン・プログラムの進歩により、情報処理装置が処理可能なコンテンツの種類および量が飛躍的に増大している。このような膨大な量のコンテンツにアクセスするために、リレーショナル・データベースや、オブジェクト指向データベースといったデータベースが利用され、クライアント・コンピュータ（以下、単にクライアントとして参照する。）によるコンテンツの検索を可能とさせている。

【0003】

データベースは、コンテンツを、マスタ・キーなどにより固有に識別して管理し、従来では、コンテンツが含むキーワードなどから生成した索引を使用して検索を実行し、検索結果としてヒットしたコンテンツまたはコンテンツ・リストをクライアントに返すことによって、クライアントのユーザに対し、コンテンツの提供を可能とする。データベースが管理するコンテンツの量が膨大でなければ、データベースが管理する全文書についてキーワード検索などの検索を実行させても、検索効率には大きな影響はないといえることができる。

10

【0004】

一方、データベースが検索しなければならないコンテンツの量が膨大になり、また、近年では検索対象とするべき索引も全文索引などが使用されるようになってきている。このため、検索効率の効率化を目的としてデータベースのコンテンツに対して区画（以下、パーティションとして参照する。）を割当て、データベースの全体を検索するのではなく、特定のパーティション内での検索の検索空間を削減することで、検索効率を向上させる試みも行われている。

20

【0005】

パーティション化は、例えば、1つのテーブルを年月など、区分すべき対象項目の範囲を区切って項目範囲ごとに同一のパーティションに格納し、パーティションの単位でコンテンツやデータの追加・削除・検索を行うことで、全期間にわたって登録されたコンテンツやデータなどを検索する場合に比較して、高速処理が可能となることが知られている。

【0006】

また、近年では、リレーショナル・データベースの登録コンテンツやデータをXMLなどの構造化文書として登録する技術も知られている。リレーショナル・データベースに各種のデータを登録する際にXMLといった構造化文書を導入することにより、従来の索引検索技術の他、DOM(Document Object Model)やSAX(Simple API for XML)などのXMLパーザを併用した検索も可能となり、いわゆるXMLデータベース（以下、XMLDBとして参照する。）も提案され、より効率的なコンテンツ検索が可能とされている。さらに、XMLDBを使用することで、リレーショナル・データベースのスキーマを修正することなく、コンテンツやデータの登録要素を修正でき、スキーマ変更に対して柔軟性を付与することができるという利点も知られている。

30

【0007】

図17は、リレーショナル・データベース(以下、RDBとして参照する。)においてXMLを管理する場合のレコードまたはカラム構造の従来例を示す。RDBによるXMLサポートは、主に、XMLデータをRDBのカラムにマッピングして格納するタイプおよびXMLデータをテキスト・データとして一つのカラムに格納して管理される場合が多い。

40

【0008】

図17に示した従来例では、リレーショナル・データベース1710は、所定のレコードを識別するために識別値（ID）を登録するカラムと、その他の属性値を登録する1つまたは複数のカラムと、XML文書のテキストを登録するカラム1720とを含んで構成されている。XML文書のサイズが小さい場合には、カラム1720にXMLを登録するだけでも充分に対応することができる。しかしながら、現在、1つのカテゴリのデータを記述するXML文書にサイズは、ますます大きくなる傾向にあり、数Gバイトとなる例も知られている。このような場合にXML全体を対象として検索を行うことは、XMLパーザなどの処理を考慮しても効率的なものということにはならない。

50

【0009】

サイズの大きなXML文書を効率的に検索するための既存技術がいくつか存在する。例えばリレーショナル・データベース製品のDB2（登録商標）（DB2は、インターナショナル・ビジネス・マシーンズ・コーポレーションの登録商標である。）では、XML文書格納および検索のため、XDA(XML

Data Area)オブジェクト1740を生成する。XDAオブジェクト1740は、複数の領域1750、1760に分割し、XML文書に対して特定の領域を索引付ける索引オブジェクト1730を構成し、大サイズのXML文書の分割領域にアクセスさせる方法が採用されている。また、XML文書は、全体またはその部分木がそれぞれノード・テーブルとして展開され、ノード・テーブルは、例えば、索引オブジェクトなどとしても使用することができる。

10

【0010】

これまで、データベースに対してパーティションを導入する技術が種々知られている。例えば、特開平7-319923号公報（特許文献1）では、データサーバは、システムにおいて異なるプロセッサを実行し、それらの各SQLステートメントを実行し、同時にかつ並行にそれらのデータベース・パーティションにアクセスし、クライアント・アプリケーションから与えられた要求全体のアクセスタイムを低減させる、方法および装置が記載されている。

【0011】

また、特開2001-142752号公報（特許文献2）では、データベースの格納領域としてm個のデータベース格納領域が与えられた場合、データベースの一つ又は複数のデータ項目をパーティショニング・キーとし、データベースを前記パーティショニング・キーに対してハッシュ関数を適用し、 n ($m \leq n$) 個の論理的な単位であるバケットに分割し、当該与えられたデータベース格納領域数に応じて各バケットを管理するデータベース格納領域の対応を決定するハッシュマップ表と、分配されたバケットを各データベース格納領域内の格納単位であるセグメントとマッピングさせるためのセグメントハッシュマップ表によってデータベースを管理する、データベース管理方法が記載されている。

20

【0012】

また、XMLなど、タグを使用する構造化文書を使用するデータベースについては、例えば、特開2008-197815号公報（特許文献3）に記載されている。特許文献3では、データベースに登録されるべき文書に対応する元の文書が当該データベースに既に登録されている場合、登録されるべき文書のうち元の文書に対して変更が発生した箇所のデータを差分データとして抽出し、索引処理部は、変更が発生した箇所のみの索引を更新することで、索引作成コストを削減するシステムが記載されている。

30

【特許文献1】特開平7-319923号公報

【特許文献2】特開2001-142752号公報

【特許文献3】特開2008-197815号公報

【発明の開示】

【発明が解決しようとする課題】

【0013】

上述したように、これまで、データベースに蓄積されるコンテンツやデータの検索効率を向上させる目的からデータベースにパーティションを割当てる技術が提案され、また、データベースの管理を効率化する目的で、データベースに階層構造を記述した構造化文書を登録する技術が提案されている。

40

【0014】

また、RDBによりXML文書を管理する場合、図17に示したように木構造に対応付けて領域分割を行う他にも、例えば日付、勤務地、明細項目など、特定の値範囲ごとに全体のデータを分類、いわゆるパーティショニングすることも可能である。XML文書を、特定の属性値によってパーティション化することによって、特定の属性値を含むデータの検索空間を縮小でき、効率的な検索処理が可能となるものと考えられる。

50

【0015】

しかしながら、従来のデータベースに対してパーティションを割当てて検索空間を削減する手法を、そのままXMLDBに適用する場合、必ずしも効率的なパーティション方法と言うことはできなかった。すなわち、XML文書は、それ自体でデータの階層構造を有することができる。このため、XML文書に対してパーティションの対象とされる対象データを有する部分木が追加されることも十分に想定することができる。また、XMLDBが管理するデータの同一の属性値で指定される、例えば<支社name=" ">などで指定される部分木には、パーティションの対象とされるべき対象データであって、異なるパーティションに分離されるべき部分木が追加される可能性が高い。

【0016】

10

したがって、XMLDBでパーティションを行うためには、パーティションの対象とする日付、品名、顧客、勤務先その他の対象データを含む部分木を特定し、特定した部分木についてのみ対象データをパーティション範囲と比較する必要がある。

【0017】

また、単にパーティション分割しただけでは、XML文書の階層構造が失われてしまい、階層構造を有効利用した検索ができなくなることで、新たな索引データを作成しなければならないという問題点も想定される。

【0018】

すなわち、これまで、データベースに対するパーティション化の有効性およびXMLを使用するデータ構成の変更の柔軟性の両方を損ねることなく、データベースの検索性を向上させ、ため柔軟なデータ構造の変化に対応することができる技術が必要とされていた。

20

【課題を解決するための手段】

【0019】

本発明では、上記従来技術の課題を解決するために、情報処理装置は、構造化文書のパーティション化する対象データを含む単位部分木を、構造化文書内で、識別し、単位部分木を規定する開始タグと、終了タグとを対応付けた領域ペアとして識別する。同時に情報処理装置は、構造化文書中の単位部分木の記述順を示す出現カウンタを生成し、部分木の構造化文書中での位置を識別させる。なお、以降の説明は、対象データを格納するデータベースとしてリレーショナル・データベースを使用するXMLDBを用いて行うが、本発明の実施形態はリレーショナル・データベースを使用するXMLDBに限定されない。

30

【0020】

単位部分木は、パーティションを行うための特定の意味を有するテキストを含む階層構造体であり、特定の意味の属性を記述するタグ単位で規定される。単位部分木は、データベース管理者やデータベースにアクセスするユーザが、XPathなどを使用して規定することができる。情報処理装置は、パーティションを行う場合、単位部分木を登録するための値パーティションと、当該単位部分木を参照するインデックスを登録するためのベース・パーティションとを作成する。

【0021】

その後、情報処理装置は、パーザ部を起動し、属性値を定義する開始タグの位置および終了タグの位置を識別して、属性値の階層構造に対応する分割位置リストを作成し記憶領域に登録する。さらに情報処理装置は、作成した分割位置リストを検索して、構造化文書の単位部分木を認識すると、その単位部分木が開始する、構造化文書中での行位置と、当該単位部分木が終了する行位置とを対応付けて領域ペアを生成する。この領域ペアには、出現カウンタにより、構造化文書中での出現位置が対応付けられて、部分木データ・リストが作成される。

40

【0022】

部分木データ・リストが作成されると、情報処理装置は、分割位置リストを参照し、パーティションの対象となる対象データを含む分割するべき単位部分木の前記開始タグ位置および前記終了タグ位置を識別し、単位部分木および単位部分木を参照するためのインデックスを、前記構造化文書から分離して識別し、分離した単位部分木と前記単位部分木を

50

識別するインデックスとを異なるパーティションに登録することにより、パーティション分割を行う。

【0023】

単位部分木は、複数の開始タグを含み、インデックスは、対応する単位部分木の最初の開始タグから生成され、また、パーティションへの登録に際して、単位部分木および単位部分木を固有に識別するためのインデックスに対し、相互参照を可能とするポインタを追加して登録する。

【0024】

さらに本発明の情報処理装置は、値パーティションに登録された単位部分木を対象データを使用して検索するためのSQL処理部と、SQL処理部による検索結果を生成する検索結果生成部を含んで構成することができる。本発明では、構造化文書は、XMLで記述することができる。データベースは、リレーショナル・データベースを使用するXMLDBとすることができる。

10

【0025】

本発明はまた、上記情報処理装置を含むデータベース・システム、情報処理方法、およびプログラムを提供することができる。

【0026】

本発明によれば、構造化文書の階層構造による検索効率を損なうことなく、パーティション化による検索空間の削減を可能な、情報処理装置を含むデータベース・システム、情報処理方法、およびプログラムを提供することができる。

20

【発明を実施するための最良の形態】

【0027】

以下、本発明を実施形態を以て説明するが、本発明は後述する実施形態に限定されるものではない。図1は、本実施形態のデータベース・システム100を示す。本実施形態のデータベース・システム100は、ネットワーク120と、ネットワーク120に接続され、ユーザにより操作されて、ネットワーク120を介してサーバ130にアクセスする複数のクライアント110、112、114、118とを含んで構成されている。

【0028】

サーバ130は、本実施形態で、データをXMLなどの構造化文書を使用してパーティショニングを実行し、格納するデータベース・アプリケーションを実装する情報処理装置として機能する。サーバ130は、特に限定されるものではないが、データベース・サーバの他、ウェブ・サーバ、SNSサーバ、メール・サーバなどの機能を併せて実装することができる。本実施形態では、ネットワーク120は、インターネットなどのネットワークを含むことが好ましいが、インターネット以外にも、WAN(Wide Area Network)、LAN(Local Area Network)などを含んで構成されていてもよい。

30

【0029】

サーバ130は、PENTIUM（登録商標）、PENTIUM（登録商標）互換チップなど、CISCアーキテクチャのマイクロプロセッサ、または、POWER PC（登録商標）などのRISCアーキテクチャのマイクロプロセッサを、シングルスコアまたはマルチコアの形態で実装することができる。また、サーバ130は、WINDOWS（登録商標）200X、UNIX（登録商標）、LINUX（登録商標）などのオペレーティング・システムにより制御することができる。さらに、サーバ130は、C、C++、JAVA（登録商標）、JAVA（登録商標）BEANS、PERL、RUBYなどのプログラミング言語を使用して実装される、CGI、サーバレット、APACHEなどのサーバ・プログラムを実行し、クライアント110～118から送付される、データ登録要求や検索要求などの各種の要求を処理する。

40

【0030】

クライアント110～118は、パーソナル・コンピュータまたはワークステーションなどを使用して実装でき、また、そのマイクロプロセッサ（MPU）は、これまで知られたいかなるシングルスコア・プロセッサまたはマルチ・プロセッサを含んでいてもよい。また、クライアント108、112、114は、WINDOWS（登録商標）、UNIX（登録商標）、L

50

UNIX（登録商標）、MAC OSなど、いかなるオペレーティング・システムにより制御されてもよい。また、クライアント110～118は、サーバ130にアクセスするため、Internet

Explorer（登録商標）、Mozilla（登録商標）、Opera（登録商標）、Netscape Navigator（登録商標）などのブラウザ・ソフトウェアを実装することができる。

【0031】

クライアント110～118と、サーバ130との間は、TCP/IPなどのトランザクション・プロトコルに基づき、HTTPプロトコルなどのファイル転送プロトコルを使用するトランザクションを使用したネットワークで接続されている。また、クライアント110～118は、サーバ130のデータベース・アプリケーションにアクセスするために、JDBC (Java（登録商標）Database Connectivity)を実装し、JDBCで定められるアプリケーションレベル・プロトコルで、サーバ130に接続してもよい。

【0032】

サーバ130は、クライアント110～118が送付した要求に対応して、サーバ130が管理するデータベースに対して要求に対応した処理、例えば受領したデータを、データベース140に格納し、さらにデータを特定の属性値に対応してパーティショニングを実行して、登録する処理を実行する。また、クライアント110～118から受領した要求が、検索要求である場合、サーバ130は、データベース140に対してSQL(Structured Query Language)を発行し、本実施形態では、パーティショニングに関連する属性を検索要求が含む場合には、パーティション単位で、XMLなどの構造化文書で登録されたデータ検索処理を実行する。

【0033】

また、パーティショニングに関連する属性を含まない検索要求の場合には、従来と同様のSQL文を使用した検索処理を実行する。なお、構造化文書としてXMLを本実施形態では例示的に説明するが、SGML、XHTML、拡張XML(Enhanced XML)など特にXMLに限定されるものではない。

【0034】

データベース・アプリケーションとしては、例えば、DB2（登録商標。）（DB2は、インターナショナル・ビジネス・マシーンズ・コーポレーションの登録商標である。）、ORACLE（登録商標）、MySQL、ACCESS（登録商標）などのいわゆるリレーショナル・データベース(RDB)を採用することができる。本実施形態のデータベース・システム100のデータベース・アプリケーションは、データベースに、パーティショニング処理を実行するためのパーティション・モジュールを拡張機能として追加実装する。

【0035】

図2は、本実施形態のサーバ装置130の機能ブロック200を示す。図2に示した各機能ブロックは、サーバ130のプロセッサが各機能部を実装させるためのプログラムをRAMなどの実行空間に読出して、プロセッサがプログラムを実行することにより、情報処理装置の機能手段として機能させることにより情報処理装置上に実現させている。サーバ130は、ネットワーク120を介してクライアント110～118からの要求を受領する。要求は、ネットワーク・アダプタ202により受領され、OS参照モデルにおける、物理層、データリンク層、トランスポート層、ネットワーク層を介してより上位プロトコルのアプリケーションに渡され、アプリケーション層を介してデータベース・アプリケーションへと要求およびデータが渡される。

【0036】

サーバ130は、本実施形態の処理を実行するため、SQL処理部210と、検索結果作成部216と、データ管理部220とを含んで構成されている。SQL処理部210は、XML文書の階層構造を指定した検索を可能とするSQLコマンドを処理し、検索結果を生成する。図2に示す実施形態では、SQL処理部210は、SQLパーザ212と、XMLパーサ214とを含んでいる。SQLパーザ212は、受領した検索要求に対応するSQLコマンドを解析し

、SAXなどを含んで実装されるXMLパーザ214による検索を可能とする。

【0037】

検索結果作成部214は、データベース140からXMLパーザ214が抽出した結果を取得して、値パーティション検索結果として表示するためのフォーマットを、例えば、XML文書またはテーブルなどの適切なフォーマットで作成し、ネットワーク・アダプタ202を介してクライアントに返却する処理を実行する。

【0038】

さらにサーバ130は、データ管理部220を備えている。データ管理部220は、本実施形態に従い、データベース・アプリケーションの機能拡張モジュールとして実装でき、パーティション生成部222と、XMLパーザ224と、XML分割部226と、分割XML登録部228とを含んで実装することができる。パーティション生成部222は、データベース140の管理上、データベース管理者がパーティションを設定する対象データ、例えば、日付、顧客名、商品名、勤務地、所属などの値について設定される範囲を使用してパーティションを生成する処理を実行する。

【0039】

また、データベース140が、例えばMySQL、ACCESSなど、プライベートレベルでデータの作成などでアクセスが許可される場合では、ユーザが自己の管理すべきデータの対象データに対応するパーティションを作成するため、例えば、「パーティション作成」要求を発行し、サーバ130が、当該要求を受領して、要求されたパーティションを作成することもできる。

【0040】

なお、データベース管理者やユーザが発行するパーティション作成要求には、パーティション分割するXML文書の部分木を、パーティション分割の粒度を規定するために記述することができる。以下、パーティション分割する粒度となる部分木を、単位部分木=Pとして参照する。単位部分木は、例えば $P = * / a / b / c \dots$ として規定することができる。なお、「*」は、ワイルドカードであり、a、b、cは、単位部分木内で属性および属性値が定義されたタグを意味する。

【0041】

また、XMLパーザ224は、本実施形態では、パーティションの対象とする構造化文書の構造解析を実行するパーザ部として機能し、処理対象とするXML文書をクライアント110～118またはデータベース140から取得し、パーティションごとにXML文書の単位部分木を分割するためのプリプロセッサとして機能する。XMLパーザ224は、この目的のため、XML文書中で、ノードの属性を規定するタグを含む行位置を識別して分割位置リストを生成し、RAMなどの適切な記憶領域に格納する。なお、XMLパーザ224についても、DOMまたはSAXを含んで実装することができ、その機能が異なるためにXMLパーザ214と別の機能部として説明するが、ソフトウェア上の構成としては単独のXMLパーザを利用して機能させることができる。

【0042】

さらにデータ管理部220は、XML分割部226と、分割XML登録部228とを含んで構成されている。XML分割部226は、XMLパーザ224が取得した、処理対象のXML文書の分割位置リストを使用して、処理対象のXML文書を、単位部分木を粒度としてパーティションに分割して行く。また、XML分割部226は、XML文書のパーティションに分割された単位部分木の元のXML文書における階層構造を指定する情報を登録するため、単位部分木を識別するためのインデックスを生成し、XML文書および単位部分木から分離する処理を実行する。

【0043】

以下、日付など対象データごとに分類されたパーティションを、値パーティションとして参照し、値パーティションに分類登録された単位部分木の元のXML文書に紐付けするインデックスを登録するパーティションを、ベース・パーティションとして参照する。なお、ベース・パーティションには、XML文書がパーティション対象となる部分木を含む場合

10

20

30

40

50

、当該パーティション対象外の部分木も登録される。また、単位部分木およびインデックスを該当するパーティションに登録する際に、

【0044】

分割XML登録部228は、XML分割部226が分割した結果を使用して単位部分木およびインデックスなどを対応するパーティションに登録する。分割XML登録部228は、元のXML文書における階層レベルと、パーティション分割された単位部分木との間の階層構造を相互参照することを可能とするため、固有の相互参照識別値をそれぞれベース・パーティションに登録すべきインデックスおよびパーティション分割すべき単位部分木に対してそれぞれ追加する。相互参照識別値についてはより詳細に後述する。

【0045】

図3は、本実施形態のサーバ130が処理対象とするXML文書300の実施形態を示す。図3に示したXML文書300は、所定の企業の受注データであり、支社ごとに、支社情報と、受注識別値と、顧客情報と、受注内容を記述する明細とが、各タグによって識別される階層構造を有するXML文書として記述されている。XML文書300の<支社name="A">で指定される親ノードには、複数の、<受注伝票>タグで指定される子ノードが含まれており、図3に示す例示的な実施形態では、<支社name="A">の属性が宣言された親ノードには、子ノード310と、子ノード320とが含まれているのが示されている。

【0046】

なお、図3に示した実施形態は例示的なものであり、親ノード<支社name="A">については、さらに多くの子ノードが記述されていてもよく、対象とするXML文書は、数万行からなるデータを含んでいてもよい。図3に示されるように、各子ノードは、<日付>タグが含まれている。これらの対象データを記述するタグはデータタグであり、属性を定義するタグを、属性定義タグとして参照する。説明する本実施形態では、パーティションの対象データは、日付であり、例えば日付に対して会計年度などの範囲を割当て、値パーティションに分類するものとして説明する。また、属性定義タグは、<支社name="A">、<受注伝票 番号="0001">、<明細 行="1">などを意味する。なお、パーティションを生成する範囲は、会計年や、出版年の他、アメリカ、日本、中国などの地理的範囲でもよいし、「a-d」、「e-f」などの文字範囲でもよい。

【0047】

図4は、本実施形態のXML文書300を、木構造400として示した図である。図4に示すように階層構造400は、「支社A」を親ノード410とし、「受注伝票0001」ノード420および「受注伝票0002」ノード430が、それぞれ支社Aノード410の子ノード420、430として含まれている。図3に示すように、日付データをパーティションの対象とする場合、子ノード420と、子ノード430とが異なる会計年度のパーティション470、480に分割されることになる。

【0048】

図4に示すように、パーティション対象の対象データを含む複数の子ノードが存在する場合、通常では、子ノードごとに対象データの値が異なる。したがって、パーティション対象の子ノードを階層構造の中から分割し、分割単位で、値パーティションに登録する処理を実行することが必要となる。このため、従来のリレーショナル・データベースで行われているように、単位部分木の単に対象データを検出し、パーティションに分類する処理では、元のXML文書の階層構造が失われ、XMLDBの有効性が損なわれてしまうことになる。このため、値パーティション470、480に含まれる子ノードの、XML文書300内でのXPathを指定することが必要となる。

【0049】

本実施形態では、XML文書300から値パーティション470、480にパーティション分割された子ノードを記述する部分木に対し、元のXML文書の階層構造を指定するベース・パーティション460を生成させ、パーティション化による検索効率の向上を、XMLDBの有効性を損なわずに可能とする。ベース・パーティション460は、元のXML文書での子ノードの階層構造におけるレベルを維持させ、かつ、特定の子ノードに対応する単位部

10

20

30

40

50

分木の参照を可能とする。

【0050】

図5は、本実施形態のサーバ130が実行するXMLDBに対する処理の概略的なフローチャートを示す。図5の処理は、ステップS500から開始し、データベース管理者またはクライアント110～118からのパーティション要求指令を受領して開始する。ステップS501で、サーバ130は、指定されたデータの範囲に対応する値パーティションを作成する。ステップS501のパーティションの作成は、リレーショナル・データベースのDDL(Data

Definition Language)を、XMLに対応可能に処理できるように拡張したSQLスクリプトを使用して作成することができる。

10

【0051】

ステップS502では、処理対象とするXML文書を読出して構文解析を実行し、XML文書から単位部分木を分割後、パーティションする単位部分木をその出現順に対応付けた部分木データ・リストを作成する。その後、ステップS503では、リストした単位部分木についてポインタを埋込んでパーティションに登録する。なお、ポインタの埋込みは、単位部分木の開始タグに、例えば隠し属性として埋め込むことができる。また、他の実施形態では、索引などに使用されるノード・テーブルの当該部分木に対応する値を格納するレコードに追加することができる。ポインタおよびポインタの埋込み処理については、より詳細に後述する。

【0052】

本実施形態のXMLのパーティション分割の処理は、ステップS504で終了する。なお、ステップS502～S504の処理は、リレーショナル・データベースにすでに格納されているXMLを対象とすることもできるし、サーバ130に対して新たな登録を要求して送付されたXML文書に対しても適用することができる。

20

【0053】

図5のステップS504からは、XMLDBに対する検索処理であり、ステップS504で、サーバ130がクライアント110～118から、検索要求を受領する。ステップS505で、検索要求にパーティションを生成した対象データが含まれているか否かを判断し、パーティションを行った対象データが含まれると判断した場合(Yes)、ステップS507で、検索要求で指定される対象データを分類したパーティションのみを検索し、ステップS508で、検索結果を検索要求を発行したクライアントに返し、ステップS509で処理を終了する。

30

【0054】

一方、ステップS505で、パーティションした属性値を含まない検索要求であると判断した場合(No)、ステップS506でパーティションにわたって検索を実行し、ステップS508でパーティションにまたがった結果をマージして検索結果を作成し、検索要求の発行元のクライアントに返し、ステップS509で処理を終了する。なお、ステップS506で、値パーティションの個別検索の他、ベース・パーティションの検索も必要な場合、ステップS508で、値パーティションの検索結果と、ベース・パーティションの検索結果とを個別にマージして検索結果を生成することもできる。

40

【0055】

なお、パーティション後の検索結果のマージは、従来のリレーショナル・データベースのパーティション分割されたデータの検索結果のマージ処理を実行する処理を使用して実装することができる。

【0056】

図6は、本実施形態において、サーバ130が、図5のステップS501でパーティションを生成する処理を実行するためのDDL文600の実施形態を示す。DDL文600は、「CREATE TABLE」により支社受注管理のテーブルを作成することを指令し、当該支社受注管理のテーブルを固有に識別するIDを指定し、対象とするXMLの属性を指定した後、「PARTITION

50

BY RANGE」によりパーティショニングを指令する。図6に示したDDL文600では、「PARTITION BY RANGE」コマンドの項目引数として、XMLのXPathを指定することか可能とされており、図示する実施形態では、読み込んだXML文書のXPathのパーティション分割するための必要なXPath深度が指定される。DDL文600は、読み込んだXML文書を、XML文書に登録された<受注伝票>から開始する子ノードのうち、最古のデータから、2006/12/31までの値パーティションと、2007/01/01から2007/12/31までの値パーティションと、2008/01/01から開始し、2008/12/31で終了する値パーティションの3つの値パーティションを生成する。

【0057】

図6に示したDDL文600には、ベース・パーティションを作成する明示的なコマンドは記述されていない。ベース・パーティション作成は、XML文書のレンジ分割のXPathが、図6中、(INFOXML/支社/受注伝票/日付)で設定されると、ベース・パーティションに格納すべきXMLの部分木の断片、すなわち、どの日付パーティションにも入らない部分木およびパーティション分割した単位部分木のインデックスが、分割アルゴリズムによって決定される。レンジ分割サーバ130は、PARTITION BY RANGE (INFOXML/…)の引数設定を識別した段階で、ベース・パーティションを登録するために、XMLDB中のデータ保管構造を定義しておくだけでよい。

【0058】

なお、PARTITION

BY RANGEおよびパーティション分割の処理は、例示的な目的で、データベース140としてDB2を使用する場合について説明するものであり、他のリレーショナル・データベースで採用される各種の適合形式に対して適宜変更することができることはいうまでもないことであるし、特定のリレーショナル・データベースの実装形式またはXML分割処理の実装上の都合に応じて、ベース・パーティションを生成する明示的なコマンドを採用することもできる。

【0059】

なお、タグのうちのどのタグの階層までをパーティション設定に利用するかについては、値パーティションの定義と直接関連し、本実施形態では、DDL文600の引数として設定された、(INFOXML/支社/受注伝票/日付)の深度に対応するタグ、具体的には、<日付>タグをパーティション設定に指定することができる。なお、DDL文で規定するXPathの深度は、値タグを明示的に記述する他、「*」などのワイルドカードを使用して暗示的に指定することもでき、深度およびXPathの階層構造に一致している限り特に限定されない。

【0060】

図7は、本実施形態のXML分割を実行する、図5のステップS502のより詳細なフローチャートを示す。図7の処理は、ステップS700で、図5のステップS501から制御を渡されて開始し、ステップS701で、XML分割位置の算出を、XMLパーザを呼び出して実行する。ステップS701の処理は、より詳細には後述するが、XMLパーザ224がXML文書を読み込んで、パーティションするための子ノードを識別するための分割位置を計算し、当該分割位置を指定する分割位置リストを生成する処理である。生成された分割位置リスト710は、RAMなどの適切な記憶領域に格納され、パーティション分割処理が終了するまで、サーバ130が保持する。

【0061】

ステップS702では、分割位置リスト710を参照して、XML文書を構成する子ノードのうち、パーティションに分類する対象データを有する単位部分木に分割して、当該単位部分木を識別するための領域ペアおよび出現カウントを生成し、部分木データ・リストの生成および単位部分木の分割、インデックス生成処理を実行する。その後、ステップS703では、生成された単位部分木のパーティション対象の値を含むタグおよびベース・パーティションに登録すべきインデックスに対して埋め込むためのポインタを、それぞれ生成する。なお、生成するポインタは、データベース140内でインデックスおよび単

10

20

30

40

50

位部分木を固有に紐付けるための値することができる。

【0062】

この目的で、ポインタとして例えば、単位部分木に対し、インデックスを固有に識別するインデックス・ポインタを埋込み、インデックスには、単位部分木を固有に識別する子ノード・ポインタを埋込むことができる。なお、インデックス・ポインタは、ベース・パーティションにおけるインデックスの登録番号、格納アドレスなどを使用することができる。また、単位部分木についても、当該値パーティション内での登録番号や格納アドレスなどを使用することができる。

【0063】

ステップ703のポインタの埋込みを実行する場合には、ステップS702のXML分割処理によってインデックスおよび単位部分木が生成されるごとに登録番号や、格納予定アドレスなどを割当てておき、各ポインタを対応付けたポインタ対応付けテーブル720を生成し、ベース・パーティションおよび値パーティションへの登録の際、ポインタ対応付けテーブル720を参照しながら並列的に登録する処理を使用することができる。

【0064】

また、ポインタ対応付けテーブル720は、値パーティションに登録すべき単位部分木が特定できた段階で、インデックスおよび単位部分木に登録すべき最新のポインタを対応付けたリストとして構成することもできる。ポインタ対応付けテーブル720は、ポインタ埋込み処理が終了した時点でメモリから削除することもできるが、検索処理のための索引データとして使用する目的で、「CREATE TABLE」で作成したテーブルの索引として、図6のIDに紐付けして、処理期間中、キャッシュしておくことができる。

【0065】

図7の処理は、ポインタ生成・埋め込み処理を終了して、ステップS704に到達すると、図5のステップS503に制御を渡し、処理を終了する。

【0066】

図8は、ステップS703の処理XML分割位置算出処理の詳細なフローチャートである。図8の処理は、図7の処理が呼出されることにより起動され、ステップS800から開始し、ステップS801で、 $i=0$ 、 $Path=／$ 、 $Start=／(0)$ 、 $End=-1$ 、 $Count=φ$ 、 $CutRangeSet=φ$ 、 $isPrevEnd=false$ を初期化する。なお、上記各変数は、下記表1の通り定義する。

【0067】

【表1】

D: XML 文書

P: レンジ分割 XPath

出力:

CutRangeSet: XML 分割位置の集合であり、分割位置リストに相当するデータ構造体。各要素は(begin, end)の形式でタグの開始位置番号と終了位置番号を保持。

要素は begin の昇順に並べる。例: CutRangeSet={ (3, 6), (7, 10), (12, 17) }

フローチャート内で用いる記号の説明:

i: D 内で現在読んでいるタグの位置番号

Path: D 内で現在読んでいるタグの XPath(スタックとして実装)

Start: 分割開始位置候補の、タグの位置番号つき XPath(スタックとして実装)

End: 分割終了位置候補のタグの位置番号。まだ候補が見つかっていなければ -1。

Count: P の出現回数を、XML の階層構造つきで保持するスタック

isPrevEnd: 直前に読んだタグが、終了タグ (</root>, など)であれば true, そうでなければ

【0068】

ステップS801で、各変数の値を初期化した後、ステップS802で、処理対象のXML文書にまだ読み込むべき開始タグまたは終了タグが存在しているか否かを判断し、読み

込むべきタグが残されていない場合（No）、XML文書の最後まで処理を進めたものとして、ステップS807で処理を終了する。

【0069】

一方、ステップS802で、処理対象のXML文書Dにまだ読み込むべきタグが存在していると判断した場合（Yes）、ステップS803で、タグ名を変数Tagに読み込み、Path=Path+“/”+Tagを計算し、タグの位置番号を1インクリメントする。ステップS803の処理で、特定の開始タグから、終了タグまでの間のXPathが生成されて行く。ステップS804で、変数Tagの値が、開始タグか否かを判断し、開始タグである場合（Yes）、ステップS806で開始タグ処理を実行する。また、ステップS804で、変数Tagの値が開始タグではない場合（No）、ステップS805で、終了タグ処理を実行し、ステップS805およびS806の各タグ処理が終了すると、処理をステップS802に戻して、XML文書Dの全タグについて処理を終了するまで、処理を反復させる。

10

【0070】

図9は、図8のステップS805およびステップS806の開始タグ処理および終了タグ処理の詳細なフローチャートを示す。ステップS900からステップS908までが開始タグ処理であり、ステップS910～ステップS917までが終了タグ処理である。まず、ステップS900～S908の開始タグ処理から説明する。

【0071】

開始タグ処理は、ステップS900でステップS804から制御を渡されて開始し、ステップS901で、Count=Count+(0)でcountスタックに0を登録する。ステップS902で、Path=Pか否かを判断し、現在読んでいるDのタグのXPathが、値パーティションに分割する単位であることをPath=Pか否かにより判断し、Path=Pでない場合（No）、処理をステップS904に分岐させ、ステップS904で、現在保持する分割位置候補を出力すべきかどうかを、「Countの先頭要素が1以上で、かつEnd $\geq$ 0」か否かの判断を使用して実行する。ステップS904で、Countの先頭要素が0かまたはEnd $\geq$ 0ではない場合（No）、ステップS906に処理を分岐し、変数End=-1、すなわち終了タグが終了候補であるか否かを判断する。End=-1である場合（Yes）、ステップS907でStart=Path+(i)として、Start位置番号を進め、ステップS908で、制御を図8のステップS802に渡し、処理を終了する。

20

30

【0072】

一方、ステップS902の判断で、Path=Pである場合（Yes）、ステップS903で、Countのすべての要素に1を加え、ステップS904で、「Countの先頭要素が1以上で、かつEnd $\geq$ 0」か否かを判断する。この場合には、ステップS904の判断は肯定的な値を返す（Yes）ので、ステップS905で変数CutRangeSetに（Start位置番号、End位置番号）を追加し、End=-1に設定する。ステップS906では、End=-1か否かを判断する。この条件下では、ステップS906の判断は、肯定的な結果を返すので（Yes）、ステップS908で、Start位置番号を進め、ステップS908で、制御を図8のステップS802に返し、処理を終了する。

【0073】

40

一方、図8の終了タグ処理は、ステップS910で、ステップS804の判断で、変数Tag内の値が、開始タグではないと判断されたことに対応して呼び出されるステップS910から開始し、ステップS911で、Countの先頭の値が1以上であるか否かを判断し、Countの先頭の値が、1以上である場合（yes）、ステップS912で、次の開始タグの検出に対応するためにEnd=iに設定し、ステップS913でStartの上位要素を取り除き、ステップS914で、Countの上位要素を取り除く。なお、Startの上位要素を取り除く処理およびCountの上位要素を取り除く処理については、図10を使用してより詳細に後述する。

【0074】

ステップS915で、Countの先頭要素が、2以上である場合、現在判断しているTagの

50

値は、その親タグの下に既に出現済みであることを示すので、ステップS 9 1 6でCutRangeSetに(Start位置番号およびその時点で判断している変数Tagの位置番号をEndとして領域ペアとして対応付けて追加する。その後、処理は、ステップS 9 1 7から図8のステップS 8 0 2に制御を渡し、終了する。

【0075】

図10は、図9のステップS 9 1 3で実行するスタックStartに対する上位要素を取り除く処理の説明図である。スタックStartは、XPath=Pが含むタグのうち、分割すべき子ノードを識別するために必要なタグに対応して、積み重ねられて行き、分割すべき子ノードの終了タグを検出することにより、子ノードの先頭を識別する機能を有する。スタックStartは、XMLパーサ222のファーストランで、対象とするXML文書のタグを解析し、XML中の特定の行番号がどのタグを含むかを識別して、分割位置リスト710の形式で、適切な記憶領域に登録する。

10

【0076】

XML分割部226は、XMLパーサ222が生成した分割位置リスト710を読み込んで、Startに追加する。その処理のため、XML分割部224は、XMLパーサ222を再度起動してセカンドランを実行させ、まず、最初のrootタグを発見した段階で、rootタグおよびその位置番号をスタックに登録する。さらにその後、タグを検出するごとに、スタックに当該タグおよび位置番号を登録する。例えば、行番号=5まで処理すると、タグ=<c />が検出される。XML分割部224は、分割単位とするXPath=P=* / b / cを識別して、タグ<c>を検出するまで、スタックにタグおよび位置番号を追加して行く。この結果行番号=4にまで処理が進行した時、スタックStartは、値スタック1020に登録する。なお、分割単位は、単位部分木を与える属性定義タグを記述するものとする。また、分割単位は、説明する実施形態の通り、ワイルドカードを使用して記述することもできるし、rootから最初の単位部分木の最終の属性定義タグまでを記述して規定することもできる。

20

【0077】

なお、<c>が検出された場合、タグ<c>が検出された位置番号を、最上位スタックに追加する。タグ<c>に対応する終了タグ</c>が検出され、説明する実施形態では、その後タグを検出した時点で、分割対象とする単位部分木を検出したものとして、スタックStartの上位2カラム1040を取り去り、対応する直前の開始タグの位置番号を最上位スタック1050として追加し、値スタック1030を生成する。この時、変数Tagには、位置番号=6が登録されている。単位部分木は、スタックの最上位に登録された位置番号からその時点で変数Tagに登録されている値までの間の区間として設定される。説明する実施形態では、第1の領域ペア=(3, 6)として取得される。

30

【0078】

その後、値スタック1030には、検出された、タグおよびその位置番号で、同一のタグを登録するカラムの値が更新され、位置番号=8で、タグ<c>が検出されると、値スタック1060が値スタック1020と同様にして作成される。さらに処理が進行し、タグが検出されると、上述した値スタック1020に対して実行したと同様の処理を実行し、上位2カラム1070を取り去り、対応する直前の開始タグの位置番号を最上位スタック1090として追加し、値スタック1080を生成する。

40

【0079】

この処理が実行された時点で、第2の領域ペア=(7, 10)が決定される。以上の処理を、処理対象のタグがなくなるまで、または</root>タグが検出されるまで実行し、処理対象のXML文書を、単位部分木に分割して行く。分割した結果は、部分木データ・リストとして、サーバ130の適切な記憶空間に格納される。

【0080】

図11は、図9の処理による本実施形態で実装するスタックCountの更新処理の概略図および生成される値スタックの値遷移1100の実施形態を示す。図11に示すスタックに登録される値は、XPathに依存して値スタックが更新されて行き、分割対象とするXPathの最後の開始タグは、処理対象のXML文書が含む分割対象のXPathのパス・カウンタとして

50

機能する。説明する実施形態では、XPath=P=*／b／cである。サーバ130のXML分割部226は、XMLパーザ224のセカンドランの時点で、図10に示すStartを生成するのと並列的に、Countを生成して行く。

【0081】

XMLパーザ224のセカンドランで、XPathに含まれるrootタグを検出することで値スタックの登録を開始し、後続するXPathの開始タグを検出するごとに、新たなスタックを追加し、値=0を登録して行く。処理を進め、XPathの最後の要素の開始タグを検出すると全スタックの値にカウント=1を加える。この段階で値スタック1112が生成される。

【0082】

その後、さらにXMLパーザ224のセカンドランで、次の開始タグが検出されるまで、上位のスタックの値を削除して行き、XPathを構成する開始タグが検出された時点で、最下位のスタックの値として0を登録して行く。そしてXMLパーザ224の処理の進行にしたがってXPathを構成する最後の構成タグが検出された時点で再度、スタックの値に1を加算することで、値スタック1114が生成され、最下位スタックの値が、処理中のXML文書内で検出された単位部分木の数を与えることが分かる。その後同様の処理を繰り返し、行番号11番目まで達すると再度値=0のスタックが積み重ねられて行き、行番号14のタグ<c>の検出で、カウント=1が全スタックに加算され、第3の領域ペア(13、16)が出力され、最終的に行番号18の</root>が検出されると、値スタックは全部が削除され、処理の終了が検出される。図11に示すXML文書には、単位部分木が3つ存在することが判断される。

【0083】

図12は、図9～図11の処理で生成された、XML分割を行うための部分木データ・リストの実施形態を示す。図12に示した実施形態では、XML文書の対応する行との対応関係を明確にするべく、分割位置リスト710に対応付けて部分木データ・リスト1300のデータ構造を示す。図12に示されるように、XML分割部226の処理によって、分割対象のXML文書が含む子ノードを識別するための集合変数CutRangeSetには、XML文書の構造にしたがって、順次(3、6)、(7、10)、(13、16)として単位部分木の範囲が登録されて行く。

【0084】

部分木データ・リストには、開始タグ処理および終了タグ処理で生成された、StartスタックのデータおよびCountスタックの値、Endの値などが登録されていて、以後の単位部分木のXML文書内での階層構造を含めてパーティション分割する場合のデータを提供する。

【0085】

本実施形態のサーバ130は、図12のCutRangeSetに登録された行番号を使用して、ベース・パーティションに登録するべきXML文書の部分木を識別して行き、XML分割部226によるベース・パーティションおよび値パーティションにパーティションするべき部分木の分割を終了する。

【0086】

図13は、ベース・パーティションに分割する部分木を生成する処理のフローチャートを示す。図13の処置は、単位部分木の特定が終了し、図12のデータ構造が構築された後に呼出されてステップS1300から開始し、ステップS1301で、i、j、T、Baseの各値を初期化する。i、jは、単位部分木をカウントするカウンタであり、iは、行番号を意味し、kは、単位部分木の番号、すなわち、Countの先頭の値に対応する。また、Tは、単位部分木を登録するスタックであり、Baseは、ベース・パーティションに登録するXML文書の部分木を登録するスタックである。

【0087】

ステップS1302では、XMLパーザ224を呼び出して、分割位置リスト710を処理し、Dにまだ読み込むべきタグがあるか否かを判断し、処理するべきタグがない場合(N)、処理対象のXML文書に対する処理が終了したので、ステップS1313でその時

10

20

30

40

50

点までに生成されたベース・パーティションに登録すべき部分木および値パーティションに登録すべき単位部分木にそれぞれポインタを追加して、各パーティションに登録する。

【0088】

一方、ステップS1302でまだ読み込むべきタグがある場合（Yes）ステップS1303で、処理中のタグを変数Tagに読み込み、カウンタiを1だけインクリメントする。ステップS1304では、 $j > k$ または $i < b_j$ であるか否かを判断する。この処理は、単位部分木の範囲を効率的に処理するために採用される。ステップS1304の判断で、 $j > k$ または $i < b_j$ の場合（No）、ステップS1305で、 $i < e_j$ であるかを判断し、 $i < e_j$ である場合（Yes）、処理中の行位置は、特定の子ノードの範囲に含まれているのでステップS1306で、 $T = T \cup \text{Tag}$ としてスタックTに追加する。

10

【0089】

また、ステップS1304で $j > k$ または $i < b_j$ の場合（Yes）、処理中の行位置は、単位部分木に含まれないので、処理鵜をステップS1308に分岐させ、 $\text{Base} = \text{Base} \cup \text{Tag}$ として追加する。一方、 $i < e_j$ でない場合（No）には、現在処理中の行位置が特定の子ノードの範囲外となっているので、ステップS1309で、 $j = j + 1$ としてカウンタjをインクリメントし、 $\text{Base} = \text{Base} \cup \text{Tag}$ とする。この段階で、スタックBaseには、処理中の行番号のタグが登録される。その後、 $T = \phi$ に再初期化する。

【0090】

ステップS1309の処理後、ステップS1306で、 $T = T \cup \text{Tag}$ に設定する。この処理の場合、ステップS1309を経たTは、 $T = \phi$ に初期化されているので、Tには、単位部分木に対応する部分木が蓄積されて行く。ステップS1306の処理後、ステップS1307では、 $i = b_j$ か否かを判断し、 $i = b_j$ の場合（Yes）、現在処理中の行位置が、単位部分木の先頭行に対応する位置なので、なお、ステップS1309で、 $i = b_j$ または $i = e_j$ の場合、当該タグ情報を、インデックスとしてベース・パーティションの登録候補としてステップS1308で、 $\text{Base} = \text{Base} \cup \text{Tag}$ として登録し、処理を再度ステップS1302に戻し、Dの残余タグがなくなるまで処理を繰り返す。

20

【0091】

一方、ステップS1307で $i = b_j$ ではない場合（No）、ベース・パーティションに登録すべき行ではないので、処理をステップS1302に直接戻し、D中に処理するべきタグがなくなるまで処理を反復させる。以上の処理を処理対象のXML文書の、ベース・パーティションに登録すべき部分木および単位部分木として登録すべき部分木を分離することができる。

30

【0092】

XML文書の分割が終了した時点で、XML登録部228が呼出され、ポインタの追加および各パーティションへの単位部分木の登録が行われる。図14は、本実施形態のパーティション分割の実施形態を示す。本実施形態の処理によって、図3に示したXML文書300は、ベース・パーティション1410と、各値パーティション1420、1430にパーティショニングされる。本実施形態では、ベース・パーティションと、値パーティションとにXML文書を分離する際にも、本来のXML文書の階層構造を維持させるために、パーティション分割された単位部分木とインデックスとを相互参照させるため、ポインタを生成し、インデックスおよびインデックスに対応する単位部分木の行位置にそれぞれポインタを生成し、登録する。

40

【0093】

ポインタは、各パーティションごとに固有の値が設定され、ベース・パーティションのインデックスには、対応する単位部分木に割当てられたポインタが登録され、単位部分木には、ベース・パーティションのインデックスに対して割当てたポインタを登録する。ポインタを生成する処理は、種々の実施形態が採用できる。例えば、ベース・パーティションおよび値パーティションを生成する際に、各パーティションに追加された部分木の行番号などをカウントするカウンタを設けておく方法を用いることができる。

50

【0094】

この実施形態では、ベース・パーティションにインデックスを追加する場合にカウンタの値Cb(Count_base)を取得し、対応する単位部分木については、登録すべき値パーティションにおけるインデックスに対応するカウンタの値Cv(Count_value)を取得し、インデックスには、ポインタCvを追加し、インデックスに対応する単位部分木には、ポインタCbを追加する処理を実行することができる。

【0095】

また、ポインタとしては、上述したようにパーティションごとに関連付けられた値を生成して追加する実施形態の他、インデックスおよび当該インデックスにより参照される単位部分木の登録処理を行う場合に、それぞれランダムにポインタRb(Random_base)およびRv(Random_value)を生成し、それぞれ対となる行に追加してもよい。ポインタの埋込についても種々の形式を採用することができる。例えば、ポインタ埋込み処理の第1実施形態では、インデックスまたはインデックスに対応する単位部分木のXMLタグの隠し属性として、インデックスおよび単位部分木の対応するXMLタグにポインタの値を埋込むことができる。

10

【0096】

例えば、図14では、ベース・パーティション1410の<受注伝票>タグ1450に、ptr="2006:467"を、<受注伝票>タグ1460に、ptr="2007:8152"を隠し属性値として追加する実施形態である。これに対応して、ベース・パーティションのインデックスのポインタであるb:344は、単位部分木の対応する<受注伝票>タグ1470の隠し属性値に、parent_ptr="b:344"として追加されている。本実施形態では、既存のXMLDBの実装に依存することなく、また検索処理のため、追加のアプリケーションを必要とすることなく、低コストで、既存のXMLDBへの実装が可能となる。

20

【0097】

また、ポインタを埋込む第2の実施形態では、既存のXMLDBの実装構成を修正する、XMLDBのために作成するノード・テーブルのレコードに、該当するポインタの値を追加して埋込む実施形態である。この実施形態では、複数のパーティションがポインタを追加したタグを含むことを排除できるので、ハードウェア・リソースの消費という観点では負担が少ない方法である。しかしながら、既存のXMLDBのノード・テーブル登録処理についても修正を加えることが必要とされるので、XMLDBの実装構成の修正も伴うため、XMLDBの実装段階からパーティショニングを行う場合には好ましく使用することができる。

30

【0098】

図15は、本実施形態にしたがってパーティション分割されるXML文書と、パーティション分割されたベース・パーティションおよび単位部分木の各値パーティションにおける登録処理の実施形態を示す。パーティション対象のXML文書1510は、ルート・ノードである<支社 name="A">以下、パーティション属性である日付の異なる<受注伝票 番号="0001">から開始する単位部分木1512と、<受注伝票 番号="0002">から開始する単位部分木1514とを含んでいる。

【0099】

各単位部分木1512のうち、先頭タグ<受注伝票...>は、インデックスとしてベース・パーティション1530に追加される。なお、ルート・ノードのタグは、図13の処理により、分類対象のXPath=P=*/b/cに含まれないのでベース・パーティション1530に分類されている。また、値パーティション1532、1534には、パーティション対象の属性値を有するタグを含む単位部分木が登録されている。ベース・パーティション1530と、値パーティション1532、1534との間は、ポインタにより相互参照が可能とされている。

40

【0100】

クライアント110~118がサーバ130に対して検索要求を発行した場合、サーバ130は、検索要求が含むキーワードを使用して検索を実行し、検索結果作成部216が階層構造を反映した検索結果を作成する。作成した検索結果は、ネットワーク・アダプタ

50

202を介してクライアント110～118に送付される。

【0101】

本実施形態では、XMLDBの検索を実行するため、例えば、SQL関数“XMLQUERY”および“XMLEXISTS”を定義する。コマンド“XMLQUERY”は、SQLのコンテキストの中からXQuery式を実行できるようにするSQLスカラ関数である。“XMLQUERY”は、変数を、XMLQUERYで指定されたXQuery式に渡すことができる。また、“XMLQUERY”は、XMLシーケンスであるXML値を戻す。

【0102】

ここで、“XMLEXISTS”の引数は、XQuery式が1つ以上の項目のシーケンスを戻すかどうかを判別し、引数に指定されたXQuery式が空のシーケンスを戻す場合、“XMLEXISTS”は、falseを戻し、その他の場合には、trueを戻す。例えば、“bookstore”テーブルのinfoxmlの列に格納されているXML文書で、／支社／受注伝票／日付の値が、2006／12／01である／支社／受注伝票の一覧を返すSQL文は、以下の洋に記述することができる。

【0103】

【数1】

```
SELECT XMLQUERY('$i/支社/受注伝票' PASSING infoxml AS "i")
FROM bookstore
WHERE XMLEXISTS('$i/支社/受注伝票/[日付 = "2006/12/01"]'
                PASSING infoxml AS "i")
```

(1)

(より詳細な定式化については、例えば、<http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.apdv.embed.doc/doc/c0023899.htm>および<http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.apdv.embed.doc/doc/c0023906.htm>を参照されたい。)

【0104】

また、検索実行の際、値パーティション内の各日付タグを参照する場合に、日付タグにポインタが付与されていなければ、その親要素(＝受注伝票)は、ベース・パーティションではなく、同一のパーティションに存在することが識別できる。一方、日付タグにポインタが付与されていれば、パーティションされており、この結果、受注伝票の親ノードが、ベース・パーティションに登録されていることが識別でき、パーティションされていないXMLDBおよびパーティション分割されたXMLDBについて、既存のSQLコマンドを使用することができる。

【0105】

さらに、パーティショニングに使用したXPath以外のパスに対して検索条件が指定された場合は、すべてのパーティションおよび検索の必要に応じてベース・パーティションを参照し、結果をマージすることにより、従来のRDBパーティション表と同じ処理を使用して検索要求に対応することができる。

【0106】

図16は、本実施形態によりパーティションすることができるXML文書の実施形態を示す。図16に示した実施形態は、MEDLINEデータベースに登録されたXML文書の実施形態である。MEDLINEデータベースは、医療文献照会用の公共データベースであり、1600万件の文献を所蔵し、ダウンロード可能なデータはXMLファイルとされ、1XMLファイルに、通常3万件の文献が記述されている。

【0107】

図16に示した実施形態のXMLについて、例えば、本実施形態にしたがい論文誌の出版年を値パーティションの対象データとしてパーティション分割することにより、例えば過去5年分の論文だけを調べるクエリーにおいて検索対象とされる検索空間を著しく削減でき、検索速度を著しく向上させることができる。また、パーティション分割を行っても、元のXMLDBに登録された時点での階層構造が、ポインタおよびインデックスの生成により保持できるので、XMLDBの検索性を大きく損なうことなく、パーティション分割を効率的

10

20

30

40

50

に実行できる。

【0108】

また、図16に示した実施形態では、値パーティション内には、文献を単位とする部分木(MedlineCitation)でパーティションに登録されるので、文献を取得する検索においても、当該パーティションのみにアクセスすれば良く、検索効率は、著しく改善することができる。以上のように、本実施形態は、RDBのレコードをタグの単位とし、それらをまとめて単一のXMLファイルにした形式のデータであって、パーティション対象の属性を含むデータに対して検索効率を向上させることができる。

【0109】

本実施形態の上記機能は、C++、Java(登録商標)、Java(登録商標)Beans、Java(登録商標)Applet、Java(登録商標)Script、Perl、Rubyなどのオブジェクト指向プログラミング言語などで記述された装置実行可能なプログラムにより実現でき、当該プログラムは、ハードディスク装置、CD-ROM、MO、フレキシブルディスク、EEPROM、EPROMなどの装置可読な記録媒体に格納して頒布することができ、また他装置が可能な形式でネットワークを介して伝送することができる。

【0110】

これまで本実施形態につき説明してきたが、本発明は、上述した実施形態に限定されるものではなく、他の実施形態、追加、変更、削除など、当業者が想到することができる範囲内で変更することができ、いずれの態様においても本発明の作用・効果を奏する限り、本発明の範囲に含まれるものである。

【図面の簡単な説明】

【0111】

【図1】本実施形態のデータベース・システム100を示した図。

【図2】本実施形態のサーバ装置130の機能ブロック200を示した図。

【図3】本実施形態のサーバ130が処理対象とするXML文書300の実施形態を示した図。

【図4】本実施形態のXML文書300を、木構造400として示した図。

【図5】本実施形態のサーバ130が実行するXMLDBに対する処理の概略的なフローチャート。

【図6】本実施形態において、サーバ130が、図5のステップS501でパーティションを生成する処理を実行するためのDDL文600の実施形態を示した図。

【図7】本実施形態のXML分割を実行する、図5のステップS502のより詳細なフローチャート。

【図8】、ステップS703の処理XML分割位置算出処理の詳細なフローチャートである。

【図9】図8のステップS805およびステップS806の開始タグ処理および終了タグ処理の詳細なフローチャート。

【図10】図9のステップS913で実行するスタックStartに対する上位要素を取り除く処理の説明図。

【図11】図9の処理による本実施形態で実装するスタックCountの更新処理の概略図および生成される値スタックの値遷移1100の実施形態を示した図。

【図12】図9～図11の処理で生成された、XML分割を行うための部分木データ・リストの実施形態を示した図。

【図13】ベース・パーティションに分割する部分木を生成する処理のフローチャートを示した図。

【図14】ベース・パーティション1410の<受注伝票>タグ1450に、ptr="2006:467"を、<受注伝票>タグ1460に、ptr="2007:8152"を隠し属性値として追加する実施形態を示した図。

【図15】本実施形態にしたがってパーティション分割されるXML文書と、パーティショ

10

20

30

40

50

ン分割されたベース・パーティションおよび単位部分木の各値パーティションにおける登録処理の実施形態を示した図。

【図 1 6】本実施形態によりパーティションすることができるXML文書の実施形態を示す。

。

【図 1 7】従来のXMLDBのデータ格納構造を示した図。

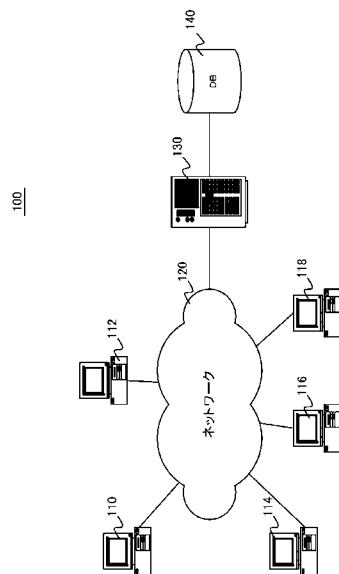
【符号の説明】

【0 1 1 2】

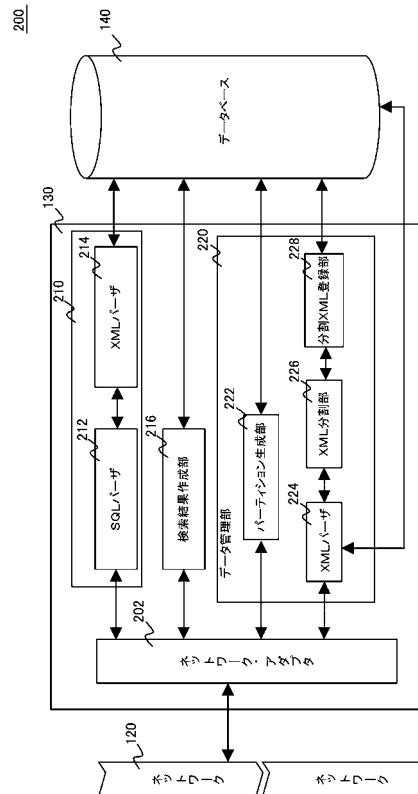
1 0 0 … データベース・システム、1 1 0 ~ 1 1 8 … クライアント、1 2 0 … ネットワーク、1 3 0 … サーバ、1 4 0 … データベース、2 0 0 … サーバ（機能ブロック）、2 0 2 … ネットワーク・アダプタ、2 1 0 … SQL処理部、2 1 2 … SQLパーザ、2 1 4 … XMLパーザ、2 1 6 … 検索結果作成部、2 2 0 … データ管理部、2 2 2 … パーティション生成部、2 2 4 … XMLパーザ、2 2 6 … XML分割部、2 2 8 … 分割XML登録部、3 0 0 … XML文書、3 1 0、3 1 2 … 単位部分木

10

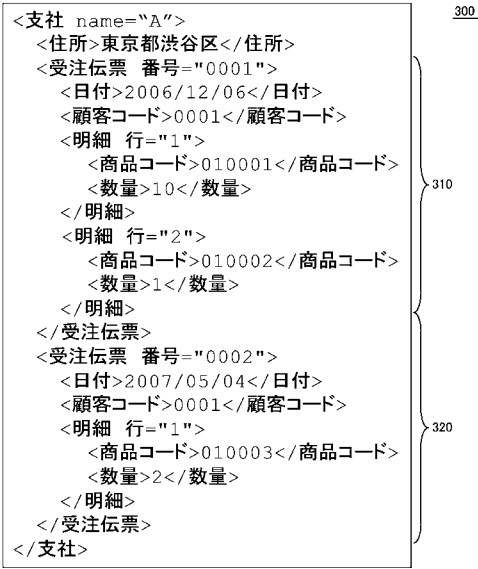
【図 1】



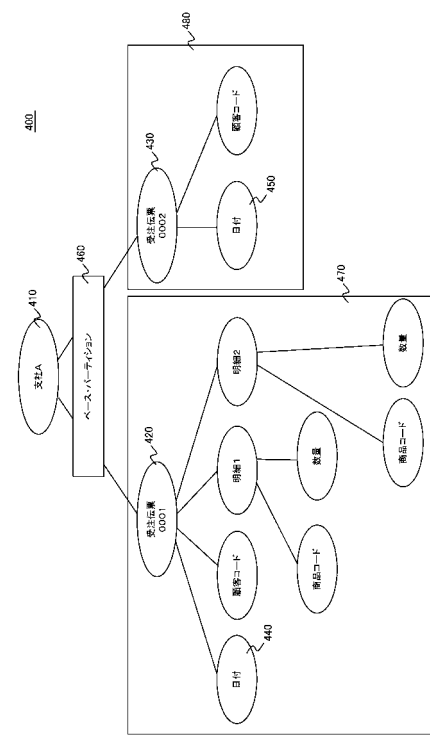
【図 2】



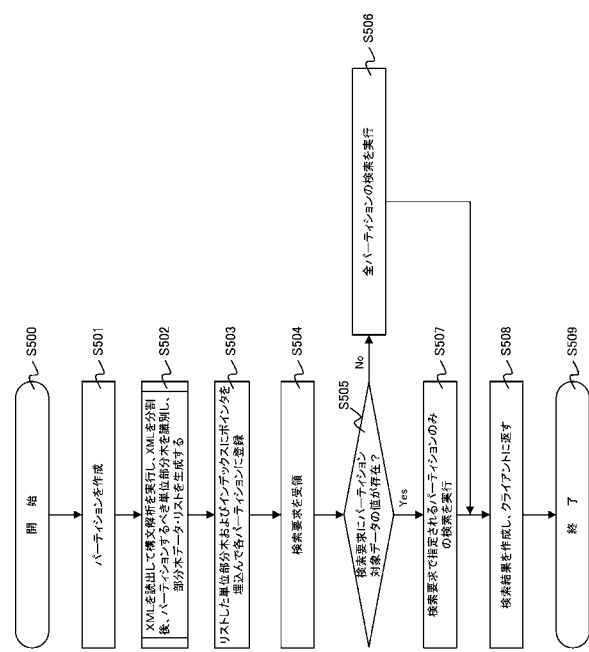
【図 3】



【図 4】



【図 5】

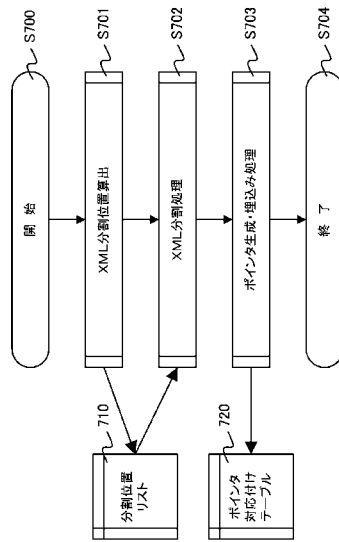


【図 6】

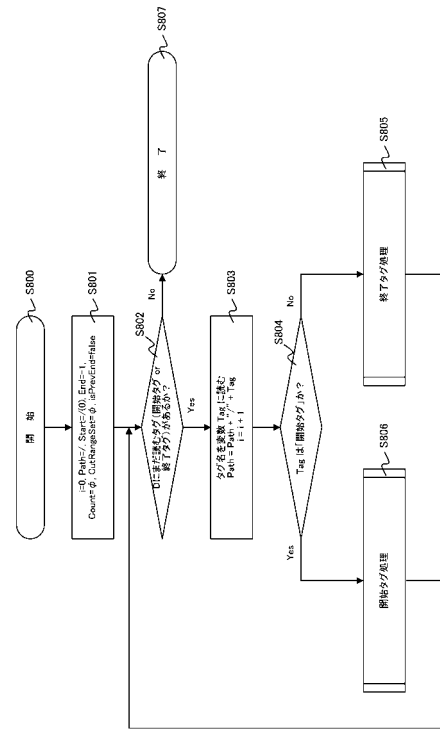
600

```
CREATE TABLE 支社受注管理 (
  ID VARCHAR(32) NOT NULL,
  INFOXML XML
)
PARTITION BY RANGE (INFOXML/支社/受注伝票/日付)
(
  PARTITION 2006 STARTING MINVALUE,
  PARTITION 2007 STARTING '2007/1/1',
  PARTITION 2008 STARTING '2008/1/1', ENDING '2008/12/31'
)
```

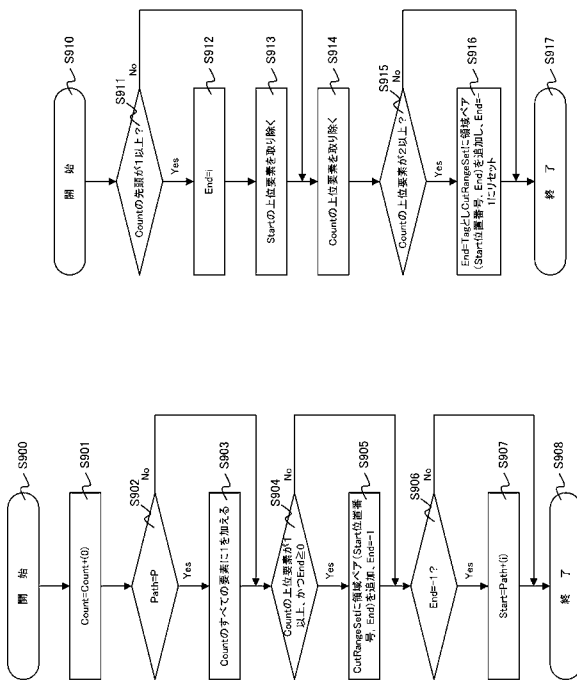
【図 7】



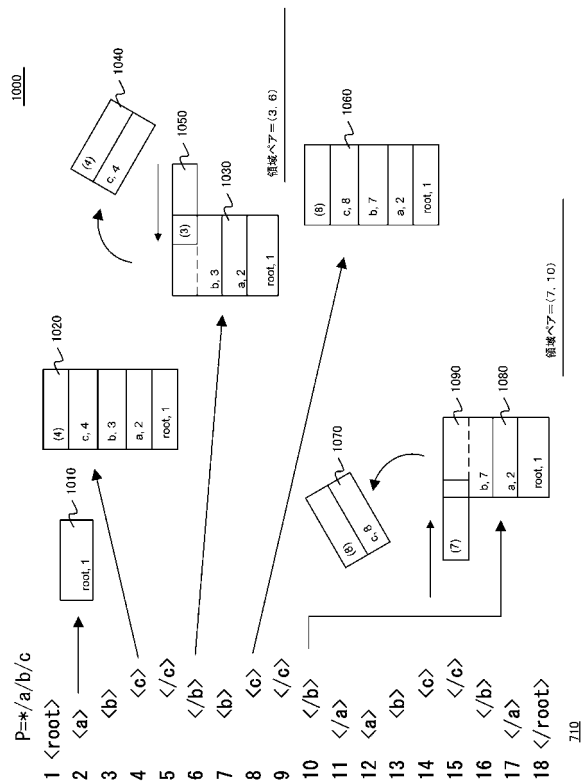
【図 8】



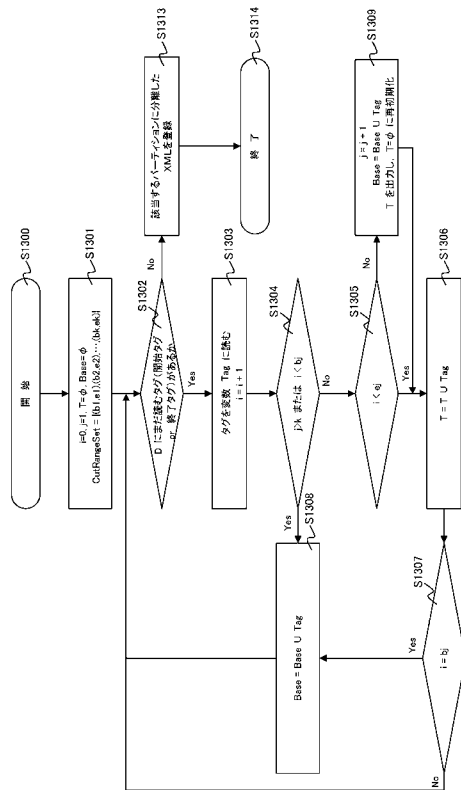
【図 9】



【図 10】

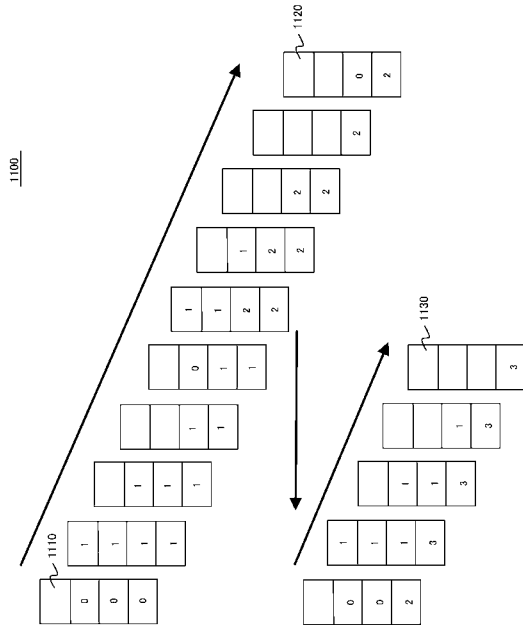


【図 1 3】



Ex. P=*/a/b/c

- 1 <root>
- 2 <a>
- 3
- 4 <c>
- 5 </c>
- 6
- 7
- 8 <c>
- 9 </c>
- 10
- 11
- 12 <a>
- 13
- 14 <c>
- 15 </c>
- 16
- 17
- 18 </root>



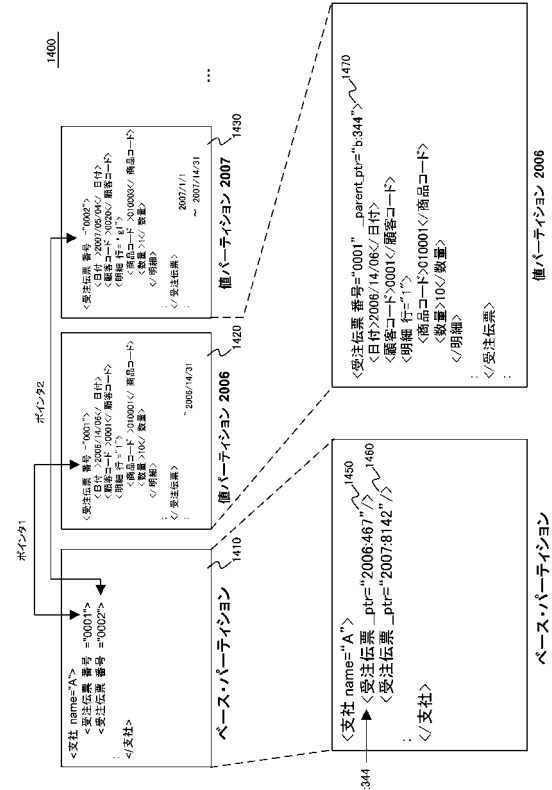
710

Ex. P=/*root/a/b/c

- 1 <root>
- 2 <a>
- 3
- 4 <c>
- 5 </c>
- 6
- 7
- 8 <c>
- 9 </c>
- 10
- 11
- 12 <a>
- 13
- 14 <c>
- 15 </c>
- 16
- 17
- 18 </root>



【図 1 4】



フロントページの続き

(74)代理人 100086243

弁理士 坂口 博

(72)発明者 新川 香

神奈川県大和市下鶴間1623番地14 日本アイ・ビー・エム株式会社 大和事業所以内

(72)発明者 吉田 一星

神奈川県大和市下鶴間1623番地14 日本アイ・ビー・エム株式会社 大和事業所内

審査官 鈴木 和樹

(56)参考文献 特開2005-135317 (JP, A)

特開2000-348038 (JP, A)

特開2003-131933 (JP, A)

国際公開第2004/081794 (WO, A1)

(58)調査した分野(Int.Cl., DB名)

G06F 17/30

G06F 12/00

JSTPlus/JMEDPlus/JST7580 (JDreamII)

CiNii